Optimización de lay-out mediante Optimización vía Simulación.

Baquela, Enrique Gabriel*, Ochoa, Joana[1]

GISOI, Facultad Regional San Nicolás, Universidad Tecnológica Nacional.

Colón 332 – CP: 2900 – San Nicolás de los Arroyos – Buenos Aires - Argentina.

ebaquela@frsn.utn.edu.ar

(1) jochoa@frsn.utn.edu.ar

Resumen

En el presente trabajo se plantea la utilización de Optimización vía Simulación (OvS) a fin de optimizar el lay-out en plantas industriales. Se desarrolló un esquema basado en el uso de R para optimizar, y un metamodelo de simulación para evaluar las soluciones propuestas. Se desarrollaron interfaces con Delmia Quest (un simulador comercial) y Simpy (un simulador de código libre implementado en el lenguaje Python).

El esquema de optimización implica la creación de un dialecto acotado para modelar el lay-out asociado a la solución a evaluar. Luego, un traductor se encarga de convertir este dialecto genérico a código fuente legible por el simulador utilizado, de manera tal de poder transformar el metamodelo a un modelo concreto. Para los metamodelos se generó una especificación a fin de poder incluir en el mismo las restricciones físicas que regulan la posibilidad de implementación o no de un determinado lay-out.

Se hicieron pruebas en varios modelos utilizando dos simuladores distintos, llegando a bueno resultados en ambos casos.

Palabras claves
Lay-Out; Optimización; Simulación

1. Introducción

Uno de los aspectos que más influye en el tiempo de ciclo de un proceso dado es el tiempo dedicado a las actividades de transportes entre unidades de producción. Este tiempo de transporte, que es tiempo no productivo, se puede reducir de varias maneras, pero en general, para un medio de transporte dado, encuentra una cota inferior que es función del lay-out de planta.

El lay-out suele ser una restricción costosa de modificar, por lo cual, ante el diseño o rediseño de una planta, es conveniente buscar el mejor lay-out posible, minimizando las distancias independientemente de la configuración de planta elegida.

El problema del lay-out se trata en la mayoría de la bibliografía mediante la minimización analítica de la distancia total a recorrer, siendo la misma una función suma de funciones individuales de distancias entre unidades productivas [1, 2]. La aproximación es muy buena cuando el proceso productivo es "lineal", en el sentido que el flujo de materiales entre unidades operativas es secuencia y único. Pero en problemas más complejos, con multiprocesos, con más de una ruta probable para el transportador del material (sea una persona o bien una máquina) este método se revela poco efectivo, al no tomar en cuenta los efectos dinámicos. En [3] se analiza el uso de algoritmos genéticos para resolver el problema, y en [4] se propone una metodología basada en el uso de algoritmos genéticos y simulación, que toma en cuenta el efecto dinámico del flujo de materiales. En esa línea, en nuestro proyecto integramos un procedimiento de Optimización vía Simulación gestionado por el software R para el tratamiento de la dinámica del problema.

2. Desarrollo

El esquema de Optimización vía Simulación utilizado plantea un modelo de dos etapas iterativas, gestionadas por un controlador [5].

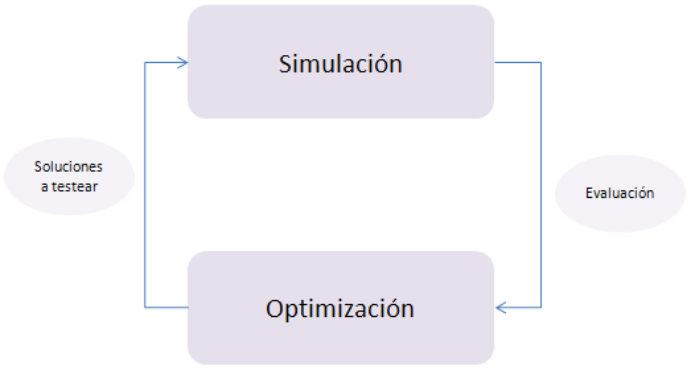

Figura 1 *Esquema de algoritmo general de Optimización vía Simulación (OvS).*

La función objetivo del problema se evalúa utilizando un simulador, el cual, implícitamente, tiene en cuenta gran parte de las restricciones del modelo. Optimizador y simulador funcionan como cajas negras independientes, pudiendo seleccionarse la combinación de algoritmo de optimización y técnica de simulación más conveniente.

Para nuestro caso, tanto el controlador como el optimizador están implementados en R[6]. La evaluación de las soluciones generadas por el optimizador se realiza en con un lenguaje de simulación apropiado, en nuestro caso Delmia Quest y Simpy[7]. Construyendo interfaces apropiadas, se podría utilizar cualquier simulador.

2.1. Modelado del problema:

El sistema a optimizar el lay-out es modelado mediante un esquema basado en dos capas. Primero, se elabora un metamodelo genérico, que contiene toda la información acerca de la interacción entre las entidades transitorias y permanentes a simular, a excepción de la asociada a localización. Es decir, se generar un modelo lógico que defina la tasa de arribo de entidades permanentes, las

capacidades de colas que no dependan del espacio físico, la secuencia de tareas de un servidor, los ruteos de clientes entre colas, los tiempos de ciclos, etc. Además, se define el dimensionamiento físico de todas las entidades involucradas en el modelo. Este modelo se genera como plantilla en el simulador a utilizar. En otras palabras, es básicamente todo el modelo de simulación sin referencias a las posiciones de las entidades.

En segundo lugar, se elabora un modelo de restricciones espaciales de nuestro sistema. Mediante el mismo, se definen las restricciones de espacio físico disponible, ubicaciones factibles para cada entidad, distancias mínimas y máximas de separación, bloqueos de tránsito, etc.

El conjunto de soluciones factibles, entonces, queda delimitado por el metamodelo de restricciones espaciales, y la función objetivo determinada por el metamodelo genérico de simulación.

2.1.1. Especificación del modelo de restricciones espaciales:

Para la definición de restricciones espaciales se definió un formalismo basado en XML. Existen dos tipos de restricciones, globales e individuales. Las restricciones globales se aplican a todas las entidades físicas del modelo de simulación. Pueden ser de dos sub-tipos:

- De localización: indican que todas las entidades deben estar localizadas dentro de esa área. Así que se define aquí el área rectangular que encierra a todas las posibles localizaciones.

- De no-localización: se refieren a áreas rectangulares que no se pueden ocupar por ninguna entidad.

Las restricciones locales presentan mayor variedad. Se aplican a una entidad individual y pueden ser de los siguientes tipos:

- De localización: la entidad tiene que estar dentro de determinada región.

- De no-localización: la entidad no puede estar en una determinada región.

- De cercanía: la entidad debería estar a una distancia igual o inferior a un determinado valor respecto de otra entidad.

- De no-cercanía: la entidad debería estar a una distancia igual o mayor a un determinado valor de otra entidad.

Las restricciones son interpretadas en forma secuencial, o sea, cualquier restricción tiene mayor peso que las restricciones anteriores, lo cual permite una forma fácil de definir una región general para posicionar equipos con subregiones prohibidas para tal tarea.

2.2. Conversión de soluciones potenciales a modelos de simulación:

La conversión de una solución potencial a un modelo para ser evaluada en el simulador se realiza por medio del controlador. La solución factible en una lista de coordenadas espaciales, cada una de las cuales corresponden con una entidad permanente del modelo a simular, por lo cual el proceso de traducción consiste en tomar la plantilla del metamodelo y modificar la ubicación de dichas entidades según como el optimizador lo requiera. Para ello, es necesario construir un conversor específico para cada simulador a utilizar.

2.2.1. Conversor Delmia Quest:

El conversor a Delmia Quest cumple la función de traducir las nuevas localizaciones desde el formato del optimizador al lenguaje BCL. El metamodelo se encuentra definido en un archivo BCL estándar (el cual incluye los reportes que hay que generar como salida) en donde las posiciones de cada entidad están referenciadas por una etiqueta. La función del conversor es simplemente reemplazar las etiquetas por los valores de la solución a evaluar.

2.2.2. Conversor Simpy:

El conversor de Simpy funciona de modo diferente. Al poder ejecutarse desde R, se accede vía memoria mediante el nombre de la entidad y se modifican sus atributos antes de iniciar una nueva simulación.

2.3. Evaluación de las soluciones:

Para evaluar la bondad de las soluciones, se realizan corridas de simulación. Se pueden aplicar dos opciones posibles:

- Simular un periodo de producción: por ejemplo, una jornada laboral completa.
- Simular ciclos en régimen estable: implica un menor número de simulaciones, pero mayor conocimiento de la capacidad del sistema. Básicamente, se arranca el sistema en marcha, y se corren pocos ciclos de producción. Mejora la performance del sistema, pero la efectividad de las soluciones halladas debería decaer por el agregado de supuestos del estado estable.

Una vez obtenido el tiempo de producción total, en caso que se violen una o más restricciones, se incrementa dicho tiempo mediante una penalización significativa a los tiempos medios.

3. Experimentos:

Para probar el desempeño de la metodología propuesta, se elaboraron 10 metamodelos de plantas fabriles, cada una con distintos números de servidores, procesos y entidades transitorias:

Tabla 1 *Listado de metamodelos*

Modelo	Cant. Servidores	Cant. Tipos Clientes	Cant. secuencias procesos
1	3	1	1
2	3	2	2
3	4	1	1
4	4	2	2
5	6	1	1
6	6	2	2
7	8	1	1
8	8	2	2
9	10	1	1
10	10	2	2

Para cada uno se testearon 3 ambientes posibles, de área cuadrada, con entrada y salida de clientes por el mismo lado, y con ninguna, una y dos regiones no válidas:

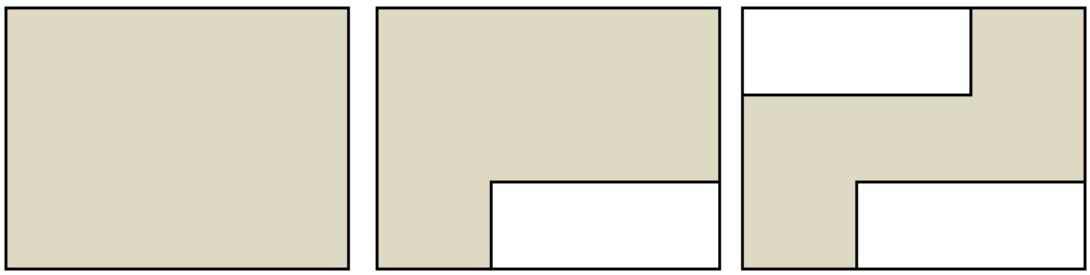

Figura 2 *Ambientes A (0 restricción), B (1 restricción) y C (2 restricciones).*

Se probó el optimizador con un algoritmo genético con tasa de mutación del 5%, población igual a 100 y cruzamiento en parejas del 50%-50%, utilizando como marco de referencia los resultados de la utilización de un procedimiento de generación aleatoria de soluciones, a fin de mensurar la bondad de las soluciones halladas. El paquete de optimización utilizado fue “genalg”. Se simuló un periodo de producción completo, asumiendo que el sistema real está activo durante 8 horas.

La comparación de las soluciones aleatorias con las generadas por nuestro algoritmo se puede observar en la siguiente tabla:

Tabla 2 *Comparación de resultados*

Modelo	Tiempo Ciclo / Tiempo Ciclo promedio	Tiempo Ciclo / Mejor tiempo de ciclo	Modelo	Tiempo Ciclo / Tiempo Ciclo promedio	Tiempo Ciclo / Mejor tiempo de ciclo
1A	0,085	1,026	6A	0,093	1,058
1B	0,025	0,959	6B	0,091	1,037
1C	0,218	1,227	6C	0,060	1,032
2A	0,025	1,147	7A	0,047	1,006
2B	0,214	1,241	7B	0,061	1,135
2C	0,128	1,064	7C	0,058	0,969
3A	0,072	1,146	8A	0,135	0,947
3B	0,114	0,957	8B	0,045	1,072
3C	0,034	1,071	8C	0,066	1,119
4A	0,133	1,228	9A	0,092	1,115
4B	0,033	0,962	9B	0,031	0,890
4C	0,044	1,105	9C	0,038	0,933
5A	0,054	1,150	10A	0,018	1,033
5B	0,105	1,126	10B	0,067	1,132
5C	0,180	0,943	10C	0,137	0,985

En la tabla anterior, se comparan las soluciones halladas por el algoritmo de OvS contra la solución promedio y la mejor solución de las generadas aleatoriamente. Como podemos observar, las soluciones se acercan mucho al mejor valor encontrado vía muestreo aleatorio del sistema.

4. Líneas de investigaciones futuras:

El trabajo presenta varias vetas para continuar ampliando el desarrollo. Algunas de ellas:

- Verificar pérdida de calidad ante modalidad de simulación de jornada completa o de ciclo.
- Analizar otros tratamientos de restricciones (no simular, por ejemplo, y descartar la solución).
- Desarrollo de un método para definir el tránsito de manera que el mismo no sea lineal.
- Considerar el estudio con equipos de transporte continuo (conveyors, power and free, agv).
- Desarrollar una aproximación robusta que se adapta a un esquema de producción variable.

5. Conclusiones

Con la potencia de cálculo de las computadoras actuales, y el relativo bajo costo de los recursos de hardware, es posible encarar problemas de optimización de forma más realista, sin necesidad de caer

en simplificaciones o casos especiales derivados de la complejidad del tratamiento analítico. Aplicar OvS al diseño de lay-out proporciona resultados interesantes, permitiendo atacar problemas con cualquier configuración del sistema de producción, y con un tiempo de modelado y recolección de datos que creemos mucho menor (a costa de un mayor costo computacional, pero debido a que el lay-out de una planta no es una decisión “de todos los días”, resulta un factor no significativo).

6. Referencias

[1] Meller, R.; Gau, K. (1996). “The facility layout problem: recent and emerging trends and perspectives”. *Journal of manufacturing systems 15, 5, 351-366*

[2] Conway, G.; Venkataramanan, M. (1994). “Genetic search and the dynamic facility layout problem”. *Computers & Operations Research 21, 8, 955-960*

[3] Kulankara, K.; Shreyes, N. (2000). “Machining fixture layout optimization using the genetic algorithm”. *International Journal of Machine Tools and Manufacture, 40, 4, 579-598*.

[4] Azadivar, F.; Wang, John. (2000). “Facility layout optimization using simulation and genetic algorithms”. *International journal of production research 38, 17, 4369-4383*

[5] Fu, M. (2002). “Optimization for simulation: Theory vs. practice”. *INFORMS Journal on Computing, 14, 192-215*.

[6] R Core Team (2012). “R: A language and environment for statistical computing”. *R Foundation for Statistical Computing, Viena, Austria. Http://www.R-project.org*

[7] SimPy Developer Team (2012). *Http://simpy.sourceforge.net/*

[8] Willighagen, E. (2012).”Genalg: R Based Genetic Algorithm – R Package version 0.1.1”. *http://cran.r-project.org/web/packages/genalg/index.html*